OIPE JC25
OCT 0 7 2002
PATENT & TRADEMARK OFFICE

AF
2100

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

#18
[illegible]
10-23-02

In re application of: **Ehnebuske et al.**	§	Group Art Unit: **2124**
	§	
Serial No.: **09/204,971**	§	Examiner: **Ingberg, Todd D.**
	§	
Filed: **December 3, 1998**	§	Attorney Docket No.: **AT9-98-267**
	§	

For: **Apparatus and Method for Performing General Integrity Checks Using Integrity Rule Checking Points in an Enterprise Application**

Certificate of Mailing Under 37 C.F.R. § 1.8(a)
I hereby certify this correspondence is being deposited with the United States Postal Service as First Class mail in an envelope addressed to: Assistant Commissioner of Patents, Washington, D.C. 20231 on September 30, 2002.

By: [illegible]
Amelia C. Nearing

RECEIVED
OCT 1 1 2002
Technology Center 2100

TRANSMITTAL DOCUMENT

Assistant Commissioner of Patents
Washington, D.C. 20231

Sir:
ENCLOSED HEREWITH:

- Reply Brief (in triplicate) (37 C.F.R. 1.192); and
- Our return postcard.

No fees are believed to be required. If, however, any fees are required, I authorize the Commissioner to charge these fees which may be required to IBM Corporation Deposit Account No. 09-0447. No extension of time is believed to be necessary. If, however, an extension of time is required, the extension is requested, and I authorize the Commissioner to charge any fees for this extension to IBM Corporation Deposit Account No. 09-0447.

Respectfully submitted,

[illegible]
Duke W. Yee
Registration No. 34,285
CARSTENS, YEE & CAHOON, LLP
P.O. Box 802334
Dallas, Texas 75380
(972) 367-2001
ATTORNEY FOR APPLICANTS

BEST AVAILABLE COPY

OIPE
OCT 0 7 2002
JC25
PATENT & TRADEMARK OFFICE

Docket No. AT9-98-267

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **Ehnebuske et al.** §
§ Group Art Unit: **2124**
Serial No.: **09/204,971** §
§ Examiner: **Ingberg, Todd D.**
Filed: **December 3, 1998** §
§
For: **Apparatus and Method for Performing General Integrity Checks Using Integrity Rule Checking Points in an Enterprise Application** §
§

RECEIVED
OCT 1 1 2002
Technology Center 2100

Assistant Commissioner for Patents
Washington, D.C. 20231

ATTENTION: Board of Patent Appeals and Interferences

Certificate of Mailing Under 37 C.F.R. § 1.8(a)
I hereby certify this correspondence is being deposited with the United States Postal Service as First Class mail in an envelope addressed to: Assistant Commissioner of Patents, Washington, D.C. 20231 on September 20th, 2002.

By: [illegible]
Amelia C. Nearing

APPELLANTS' REPLY BRIEF

This brief is in furtherance of Appellants' Appeal Brief, filed in this case on April 22, 2002.

The fees required under § 1.17(c), and any required petition for extension of time for filing this brief and fees therefore, are dealt with in the accompanying TRANSMITTAL OF REPLY BRIEF.

This brief is transmitted in triplicate. (37 C.F.R. 1.192(a))

I. Incorrect Statements of Status in Examiner's Answer

On page 2, section 2 of the Examiner's Answer, the Examiner states that Appellants' Brief does not contain a statement identifying related appeals and interferences which will directly affect or be directly affected by or have a bearing on the decision in the pending appeal. Appellants respectfully disagree and direct the Board's attention to page 2 of Appellants' Appeal Brief which clearly states that there are no appeals or interferences that will directly affect or be directly affected by the Board's decision in the pending appeal.

In addition, the Examiner, in the statement of the grouping of claims on pages 2-3, section 7, fails to include claims 14 and 15 in the grouping of claims. Appellants clearly stated in the Appeal Brief, page 4, that claim 14 belongs to group IX and claim 15 belongs to group X.

Moreover, the Examiner, on page 3, section 9 fails to identify the actual references relied upon which are part of the record but rather states "the copy of the appealed claims contained in the Appendix to the brief is correct." To aid the Board, Appellants respectfully submit that the prior art relied upon by the Examiner is Martin, Principles of Object-Oriented Analysis and Design, PTR Prentice Hall, 1993[1], McConnell, Code Complete A Practical Handbook of Software Construction, Microsoft Press, p. 94-109, 1993 and McDaniel, IBM Dictionary of Computing, pages 1, 119, 185, 586 and 597, 1993.

II. Statements of Clear Bias on the Part of the Examiner

The Examiner's inability to examine Appellants' claims in an objective manner without hindsight and a predisposition to rejecting all of the claims despite the features recited therein is clearly demonstrated in the Examiner's statement challenging the Board to allow the present application on page 26 of the Examiner's Answer which states, "If a twenty year right to exclude others from integrity checking in an object oriented programming environment is to be granted it will be granted by the Board of Patent Appeals and Interference (BPAI)." This statement clearly supports the very arguments Appellants have been asserting from the beginning of prosecution of

[1] It should be noted that the Examiner, in the PTO Form 892 Submitted with the First Office Action, incorrectly identified the publication date as June 1, 1992, when the copyright

this application, i.e. that the Examiner has generalized the invention to a point where he can allegedly read a general textbook on the generalization of Appellants' invention. The Examiner has not, and continues to refuse to, examine the actual features recited in the claims but decides to rather reject a different invention that fits better into the mold he wishes to make.

Appellants are not claiming the general concept of integrity checking in an object oriented programming environment. This fact is clear from the language of the pending claims. For example, claim 1 recites "identifying a point in a unit of work where application state integrity is to be verified, wherein the unit of work includes a plurality of participants." This feature is not part of general integrity checking in an object oriented environment. Claim 1 further recites "responsive to determining that the unit of work is to be completed, obtaining rules associated with each participant in the unit of work." This feature is also not part of general integrity checking in an object oriented environment. Claim 1 goes on to recite "responsive to obtaining the rules, running the rules obtained for each of the participants to verify the integrity of an application state, according to the plurality of participants." Again, this is not part of general integrity rule checking in an object oriented environment.

While the present invention makes use of integrity checking, the particular manner in which the present invention uses integrity checking is not simply to do so in an object oriented environment. To see that the features of the present invention are not present in integrity checking in an object oriented environment in general, one need only look at the Martin reference that the Examiner relies upon, which Appellants have shown from the beginning of prosecution does not teach or suggest any of the features of the presently claimed invention.

III. <u>Definition of Terms</u>

The Examiner continues to contest Appellants' asserted right not to be bound by the Examiner's interpretation of terms. The Examiner states, in many places in the Examiner's Answer, that Appellants "left the door open" for the interpretation of the terms "unit of work" and "participant" and thus, the Examiner is going to make up his own definition of these terms and stick to it. Appellants in no way left any door open nor did Appellants agree to any

notification clearly states 1993.

definition set forth by the Examiner.

In the First Office Action, the Examiner set forth a number of interpretations that the Examiner asserted were going to be used to limit the use of the terms in the claims. Appellants, in their response to the First Office Action, and in an attempt to aid the Examiner, offered other example definitions obtained from the specification for these terms and asserted that these example definitions were merely examples and were not intended to limit the scope of the present claims. The Examiner then asserted, and continues to assert, that Appellants are bound by their examples when Appellants clearly stated that they were only offering the example definitions as examples and were not in any way meaning to limit the claims. The Examiner then as much as states that he is a "finder of fact" and that the example definitions are now "findings of fact." The Examiner is not a "finder of fact" and the example definitions are not "findings of fact" despite the allegations made by the Examiner. The scope of the terms used in the present claims are only limited by the present specification as interpreted by one of ordinary skill in the art, not some arbitrary definition imposed by the Examiner in order to try and fit the claims into a mold formed by the Examiner's selected reference.

It is well established that the Examiner is permitted to give claims their broadest reasonable interpretation and that Appellants can be their own lexicographer. In the present application, however, the Examiner has come to the conclusion that the Examiner can be his own lexicographer in order to give recited terms a more limiting interpretation that better fits the reference rather than an interpretation that accurately defines the invention as described in the present specification. Appellants again assert that the definition of the terms used in the present claims should not be narrowly interpreted in view of the Examiner's statements.

Moreover, Appellants respectfully submit that even if the Examiner's interpretation of the claim terms were adopted by the Board, the claimed features still are not taught or suggested by the cited references, as discussed in greater detail hereafter.

IV. Taking the Martin Reference as a Whole

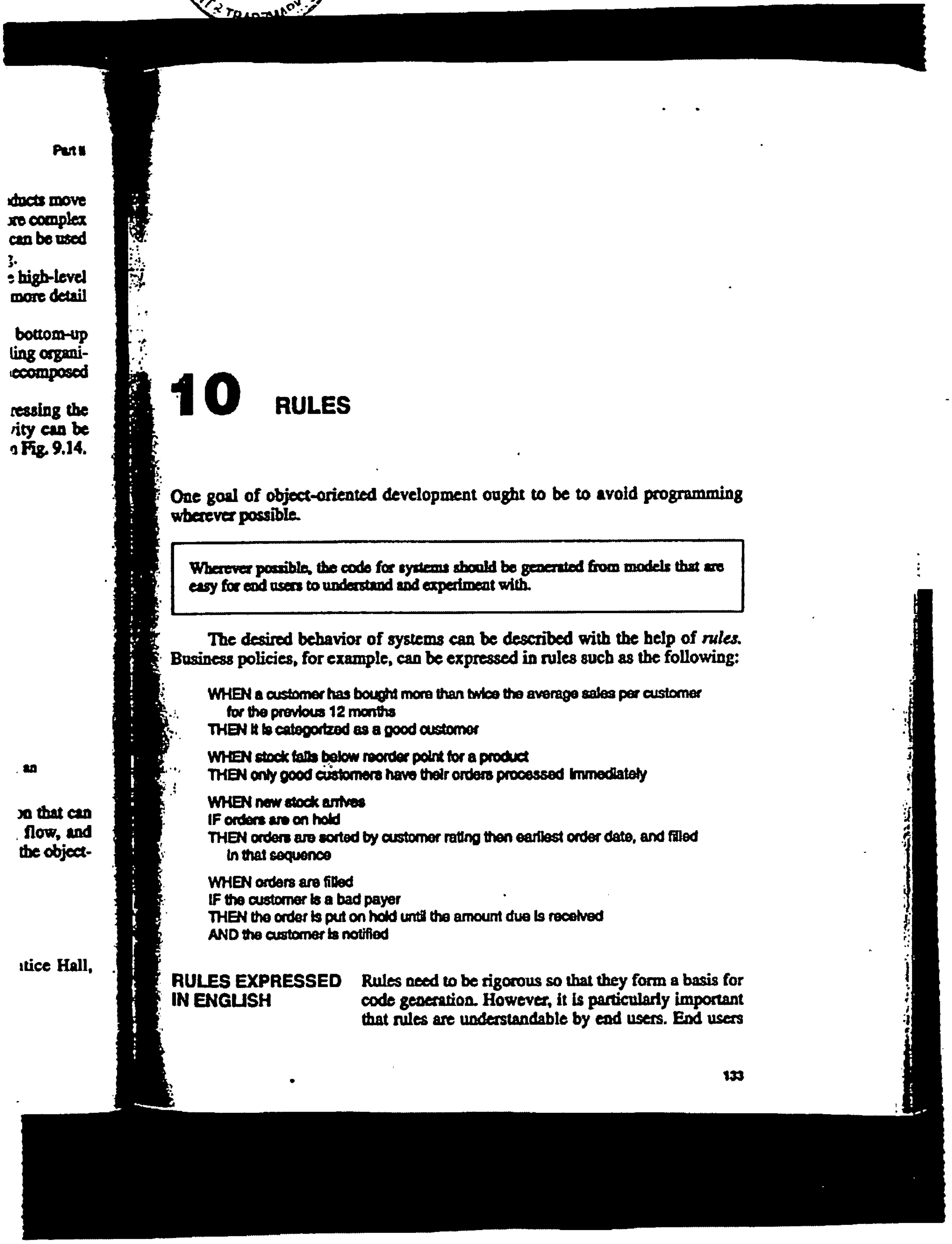
Appellants continue to maintain their objection to the Examiner's statement that the Martin reference must be taken as a whole and thus, Appellants must digest all 400 plus pages of the Martin reference and respond to the entire reference as a whole rather than just those sections relied upon by the Examiner. As has been consistently stated by Appellants, such a stance clearly shifts the burden of establishing a prima facie case of obviousness from the Examiner to the Appellant. If such a position were correct, why would the Examiner need to point to any section of Martin at all? Why couldn't the Examiner merely say that the features of the claims are taught in Martin and let Appellants discern where such features are allegedly taught? This clearly does not meet with the requirements for establishing a prima facie case of obviousness under Graham v. Deere.

Moreover, it seems that the Examiner believes that it is not too much of a burden for Appellants to review the entire 400 plus pages of Martin, or for the Board to do so, yet it is too much of a burden for the Examiner to cite those other sections of Martin that allegedly teach the features of the present claims. Appellants object to such a notion and will continue to only address those sections of the Martin reference specifically relied upon by the Examiner as alleged support for his position.

V. Sections of Martin Cited as Allegedly Supporting the Examiner's Position

The Examiner, on pages 5 and 7 of the Examiner's Answer, sets forth the sections of Martin that the Examiner is relying upon to be pages 133-144 and 147. In the following, Appellants will look at each page individually and contrast its teachings against an exemplary claim, claim 1, to illustrate how the Martin reference lacks even the basic teachings or suggestions necessary to support a rejection under 35 U.S.C. § 103(a).

Page 133 of Martin is as follows:



Part II

iducts move
ore complex
can be used
3.
e high-level
more detail

bottom-up
ling organi-
iecomposed

ressing the
vity can be
n Fig. 9.14.

. an

on that can
. flow, and
the object-

ntice Hall,

10 RULES

One goal of object-oriented development ought to be to avoid programming wherever possible.

> **Wherever possible, the code for systems should be generated from models that are easy for end users to understand and experiment with.**

The desired behavior of systems can be described with the help of *rules*. Business policies, for example, can be expressed in rules such as the following:

```
WHEN a customer has bought more than twice the average sales per customer
   for the previous 12 months
THEN it is categorized as a good customer

WHEN stock falls below reorder point for a product
THEN only good customers have their orders processed immediately

WHEN new stock arrives
IF orders are on hold
THEN orders are sorted by customer rating then earliest order date, and filled
   in that sequence

WHEN orders are filled
IF the customer is a bad payer
THEN the order is put on hold until the amount due is received
AND the customer is notified
```

RULES EXPRESSED IN ENGLISH

Rules need to be rigorous so that they form a basis for code generation. However, it is particularly important that rules are understandable by end users. End users

133

From this page all that can be ascertained is:

(1) a goal of object oriented programming is to avoid programming wherever possible;

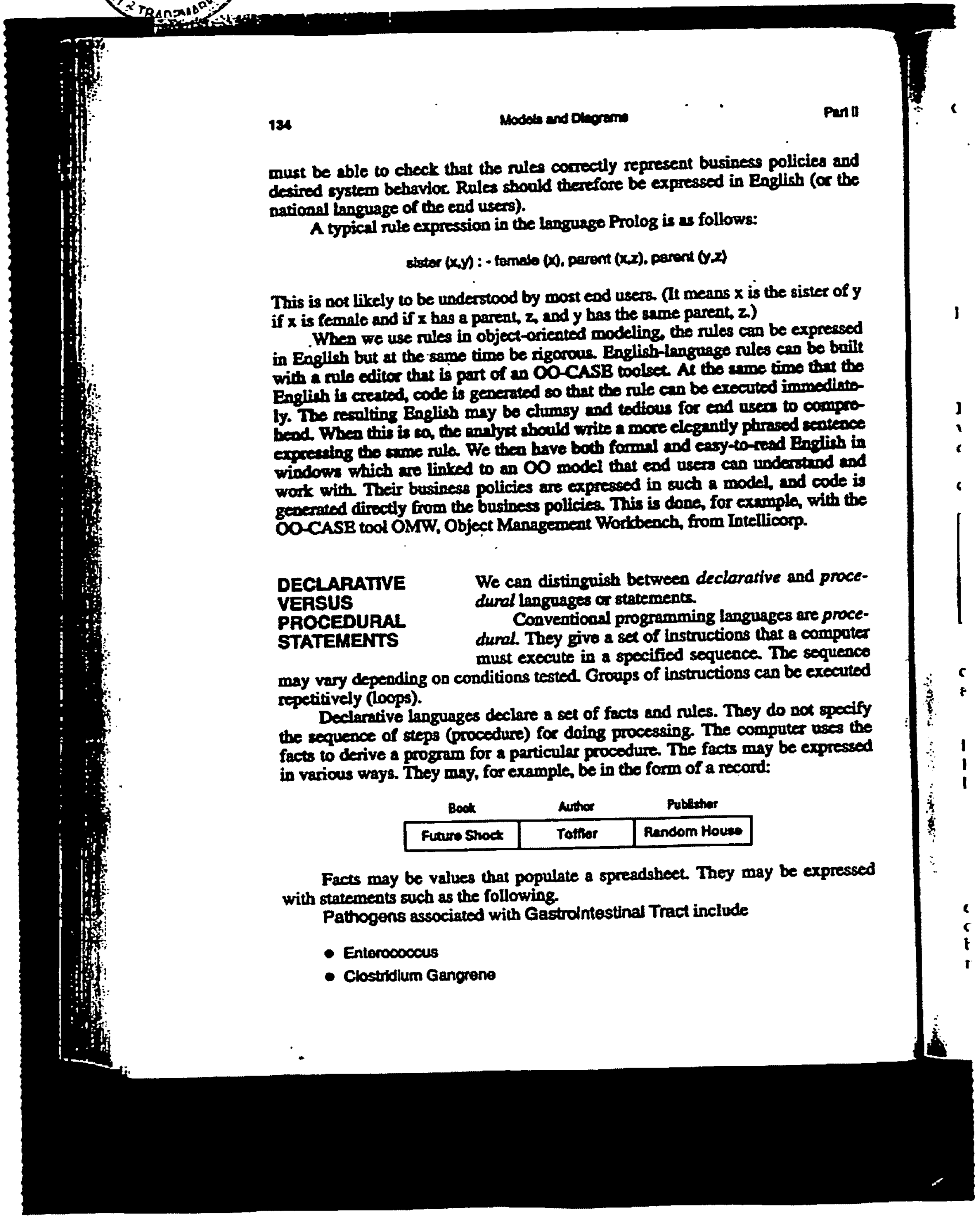
(2) that the desired behavior of systems can be described with the help of rules;

(3) that business policies can be expressed in rules; and

(4) rules need to be rigorous and understandable by end users.

There is nothing on this page of Martin that speaks to any of the features of independent claim 1. Specifically, there is no teaching or suggestion on this page of Martin regarding units of work, identifying a point in a unit of work where state integrity is to be verified, a unit of work having a plurality of participants, obtaining rules for each participant in a unit of work in response to a determination that the unit of work is to be completed, or running the rules obtained for each of the participants to verify the integrity of an application state according to the plurality of participants in response to obtaining the rules.

Page 134 of Martin is as follows:



must be able to check that the rules correctly represent business policies and desired system behavior. Rules should therefore be expressed in English (or the national language of the end users).

A typical rule expression in the language Prolog is as follows:

```
sister (x,y) : - female (x), parent (x,z), parent (y,z)
```

This is not likely to be understood by most end users. (It means x is the sister of y if x is female and if x has a parent, z, and y has the same parent, z.)

When we use rules in object-oriented modeling, the rules can be expressed in English but at the same time be rigorous. English-language rules can be built with a rule editor that is part of an OO-CASE toolset. At the same time that the English is created, code is generated so that the rule can be executed immediately. The resulting English may be clumsy and tedious for end users to comprehend. When this is so, the analyst should write a more elegantly phrased sentence expressing the same rule. We then have both formal and easy-to-read English in windows which are linked to an OO model that end users can understand and work with. Their business policies are expressed in such a model, and code is generated directly from the business policies. This is done, for example, with the OO-CASE tool OMW, Object Management Workbench, from Intellicorp.

DECLARATIVE VERSUS PROCEDURAL STATEMENTS

We can distinguish between *declarative* and *procedural* languages or statements.

Conventional programming languages are *procedural*. They give a set of instructions that a computer must execute in a specified sequence. The sequence may vary depending on conditions tested. Groups of instructions can be executed repetitively (loops).

Declarative languages declare a set of facts and rules. They do not specify the sequence of steps (procedure) for doing processing. The computer uses the facts to derive a program for a particular procedure. The facts may be expressed in various ways. They may, for example, be in the form of a record:

Book	Author	Publisher
Future Shock	Toffler	Random House

Facts may be values that populate a spreadsheet. They may be expressed with statements such as the following.

Pathogens associated with GastroIntestinal Tract include

- Enterococcus
- Clostridium Gangrene

From this page, all that can be ascertained is:

(1) rules should be expressed in English;

(2) rules can be built using a rule editor that is part of an OO-CASE toolset such that at the same time that the English is created, code is generated so that the rule can be executed immediately;

(3) there may be formal and easy-to-read English in windows which are linked to an OO model that end users can understand and work with;

(4) business policies are expressed in such a model and code is generated directly from the business policies;

(5) procedural languages specify a set of instructions that the computer must execute; and

(6) declarative languages declare a set of facts and rules and the computer uses these facts to derive a program for a particular procedure.

Again, there is nothing on this page of Martin that speaks to any of the features of independent claim 1. Specifically, there is no teaching or suggestion on this page of Martin regarding units of work, identifying a point in a unit of work where state integrity is to be verified, a unit of work having a plurality of participants, obtaining rules for each participant in a unit of work in response to a determination that the unit of work is to be completed, or running the rules obtained for each of the participants to verify the integrity of an application state according to the plurality of participants in response to obtaining the rules.

Page 135 of Martin is as follows:

- Bacteroids
- Klebsiella
- Pseudomonas
- E Coli
- Enterobacterium
- Proteus

Facts may also be expressed with equations, for example:

PRINCIPAL = INSTALLMENT X 100/INTEREST_RATE (1 –
(1 + INTEREST_RATE/100) ** – N) (1 + INTEREST_RATE/100)

If *any* three of the four variables in this financial equation are entered, the fourth will be calculated. *The user does not give the sequence of steps in doing the calculation.* Similarly, several simultaneous equations could be used.

End users, who generally have difficulty understanding programs and checking their correctness, can easily understand statements of facts and rules.

> Declarative statements are much easier to grasp and validate than procedural language.
>
> Where possible, we should build systems from declarative statements linked to OO diagrams that end users can understand.

To a large extent, but not completely, the design of an OO system can be created automatically from facts and rules of the type described in this chapter, and code can be automatically generated.

MAKING BUSINESS KNOWLEDGE EXPLICIT

The rules described in this chapter capture the know-how about how the business or system should operate.

> Rules are encapsulated business knowledge.

We need techniques with which business know-how can be captured, made explicit, made easy to read, and translated directly into executable code. With traditional structured techniques, business policies are not made explicit. They become buried in code written in COBOL or other languages. One rule is often reflected in multiple programs and may be virtually impossible to extract from

From this page of Martin, all that can be ascertained is:

(1) facts may be expressed as equations;

(2) systems should be built using declarative statements because they are more easily understood by users;

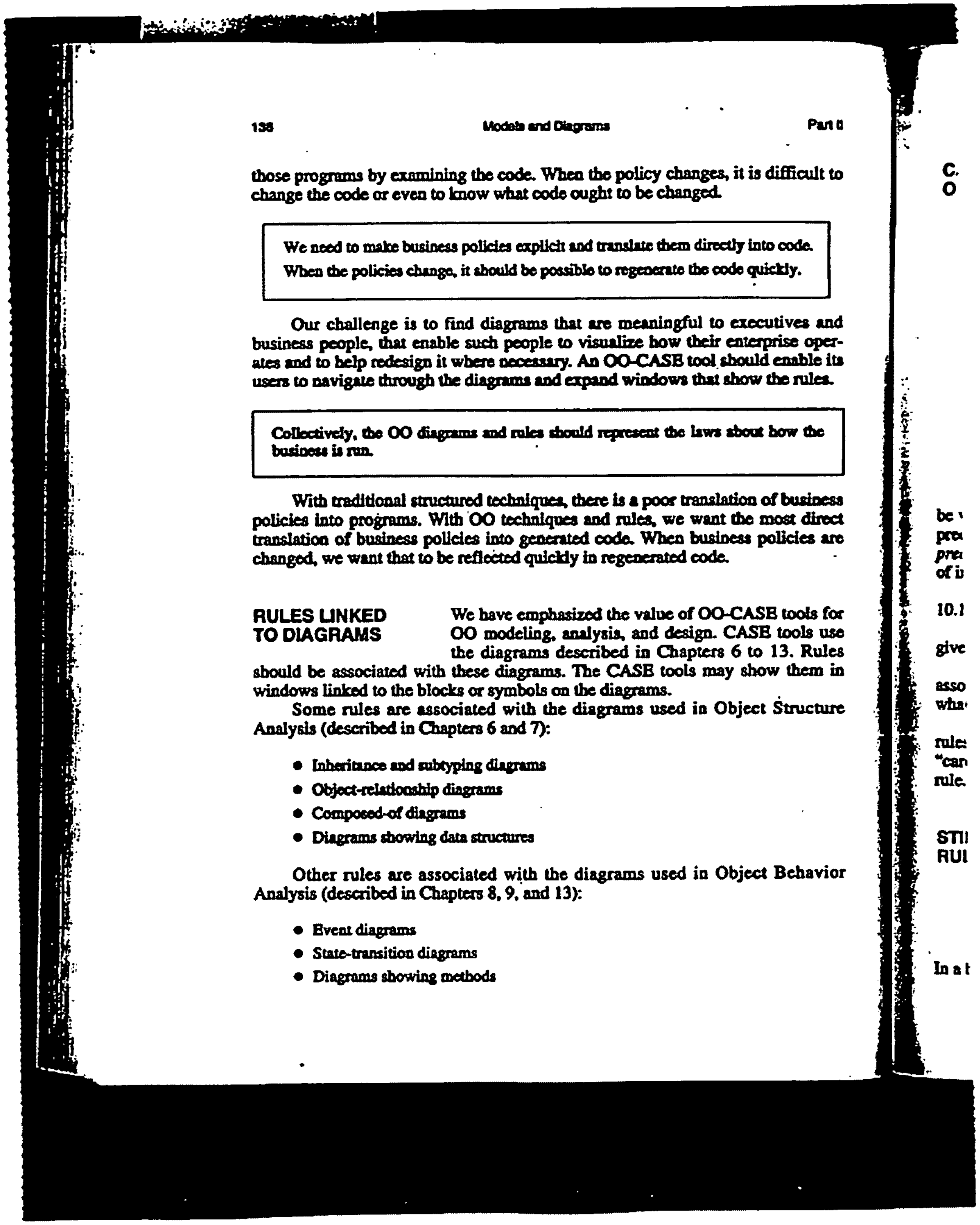
(3) an OO system may be created automatically from facts and rules and code can be automatically generated;

(4) rules are encapsulated business knowledge; and

(5) traditionally, business policies are not made explicit.

Yet again, there is nothing on this page of Martin that speaks to any of the features of independent claim 1. Specifically, there is no teaching or suggestion on this page of Martin regarding units of work, identifying a point in a unit of work where state integrity is to be verified, a unit of work having a plurality of participants, obtaining rules for each participant in a unit of work in response to a determination that the unit of work is to be completed, or running the rules obtained for each of the participants to verify the integrity of an application state according to the plurality of participants in response to obtaining the rules.

Page 136 of Martin is as follows:



those programs by examining the code. When the policy changes, it is difficult to change the code or even to know what code ought to be changed.

> We need to make business policies explicit and translate them directly into code. When the policies change, it should be possible to regenerate the code quickly.

Our challenge is to find diagrams that are meaningful to executives and business people, that enable such people to visualize how their enterprise operates and to help redesign it where necessary. An OO-CASE tool should enable its users to navigate through the diagrams and expand windows that show the rules.

> Collectively, the OO diagrams and rules should represent the laws about how the business is run.

With traditional structured techniques, there is a poor translation of business policies into programs. With OO techniques and rules, we want the most direct translation of business policies into generated code. When business policies are changed, we want that to be reflected quickly in regenerated code.

RULES LINKED TO DIAGRAMS

We have emphasized the value of OO-CASE tools for OO modeling, analysis, and design. CASE tools use the diagrams described in Chapters 6 to 13. Rules should be associated with these diagrams. The CASE tools may show them in windows linked to the blocks or symbols on the diagrams.

Some rules are associated with the diagrams used in Object Structure Analysis (described in Chapters 6 and 7):

- Inheritance and subtyping diagrams
- Object-relationship diagrams
- Composed-of diagrams
- Diagrams showing data structures

Other rules are associated with the diagrams used in Object Behavior Analysis (described in Chapters 8, 9, and 13):

- Event diagrams
- State-transition diagrams
- Diagrams showing methods

C.
O

be
pre
pre
of i

10.1

give

asso
wha

rule
"can
rule.

STI
RUI

In a t

From this page, all that can be ascertained is:

(1) business policies should be made explicit and should be translated directly into code;

(2) when the policy changes, it should be possible to regenerate the code quickly;

(3) an OO-CASE tool should enable users to navigate through diagrams and expand windows to show the rules for the diagrams which are more meaningful to executives and business people;

(4) the OO diagrams and rules should represent the laws about how the business is run;

(5) CASE tools use diagrams and rules should be associated with these diagrams which the CASE tools may show in windows linked to blocks or symbols of the diagrams;

(6) some rules are associated with diagrams used in object structure analysis; and

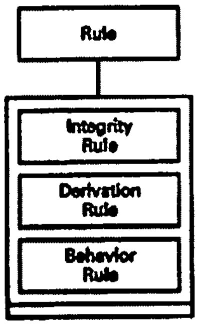
(7) other rules are associated with diagrams used in object behavior analysis.

As with pages 133-135, there is nothing on this page of Martin that speaks to any of the features of independent claim 1. Specifically, there is no teaching or suggestion on this page of Martin regarding units of work, identifying a point in a unit of work where state integrity is to be verified, a unit of work having a plurality of participants, obtaining rules for each participant in a unit of work in response to a determination that the unit of work is to be completed, or running the rules obtained for each of the participants to verify the integrity of an application state according to the plurality of participants in response to obtaining the rules.

Page 137 reads as follows:

CATEGORIES OF RULES

Several different types of rules exist. Business rules can be classified as integrity rules, derivation rules, and behavior rules [2].



Integrity rules state that something must be true. For example, a value must be within a certain range, an object relationship must have a stated cardinality, a precondition must hold before an operation is executed, and so on. *Operation precondition rules* and *operation postcondition rules* are important special cases of integrity rules.

Integrity rules should start with the phrase "it should always hold that." Box 10.1 gives some examples of integrity rules.

Derivation rules state how a value or a set of values is computed. Box 10.2 gives examples of derivation rules.

Behavior rules describe dynamic aspects of behavior. They are commonly associated with event diagrams. They often express business rules that describe what conditions must be true for an operation to be triggered.

In some methodologies, cardinality constraints are referred to as "business rules" and are the only form of business rule. These should be referred to as "cardinality rules" to indicate that they are only one narrow type of business rule.

STIMULUS/RESPONSE RULES

Stimulus/response rules express triggering behavior and generally have the form

```
WHEN    <event>
IF      <condition to fulfill>
THEN    <operation>
```

In a business model this can be

From page 137, all that can be ascertained is:

(1) business rules can be classified as integrity rules, derivation rules and behavior rules;

(2) integrity rules stat that something must be true;

(3) derivation rules state how a value or a set of values is computed;

(4) behavior rules describe dynamic aspects of behavior; and

(5) stimulus/response rules express triggering behavior.

This is the first time that Martin teaches anything remotely related to the present claims, i.e. integrity rules. The presently claimed invention verifies the integrity of an application state and Martin teaches integrity rules. However, this page of Martin still does not teach the specific features recited in claim 1. Specifically, there is no teaching or suggestion on this page of Martin regarding units of work, identifying a point in a unit of work where state integrity is to be verified, a unit of work having a plurality of participants, obtaining rules for each participant in a unit of work in response to a determination that the unit of work is to be completed, or running the rules obtained for each of the participants to verify the integrity of an application state according to the plurality of participants in response to obtaining the rules. All that is taught on page 137 of Martin is the general concept of integrity rules.

Pages 138 and 139 merely provide examples of integrity rules and derivation rules and thus, are not reproduced below. Suffice it to say, these pages of Martin do not add anything to what was taught in pages 133-137 other than to give explicit examples of rules. Even the examples of integrity rules shown on page 138 merely state "IT MUST ALWAYS HOLD THAT..." and provide no further teaching regarding the specific features of claim 1 discussed above.

Page 140 is as follows:

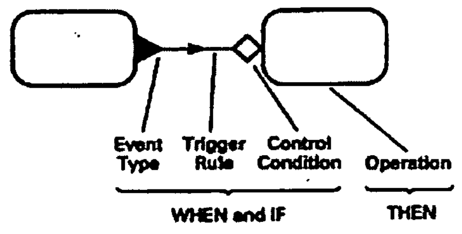
```
WHEN     an event happens
IF       a condition is fulfilled
THEN     do something
```

In a technical system, it can be

```
ON       <stimulus>
IF       <condition>
THEN     <response>
```

In these expressions the IF clause is optional.

This form of expression relates to the basic construct of the event diagram:



Such stimulus/response rules can be expressed rigorously but can also be translated into an English statement that end users can validate. For example

```
WHEN Account is posted to Sales Ledger
IF Customer is of type D
   and Balance has been negative for 18 days
THEN set Status to Credit Stopped.
```

The THEN part of a rule is an operation. For example, set Status to Credit Stopped is an operation. Often, however, the THEN part of a rule invokes an operation by name. For example, when a stock item falls below the reorder point, send a request to Reorder the stock item.

OPERATION CONDITION RULES

As commented in the previous chapter, each operation has its own context-independent precondition and postcondition. Preconditions and postconditions should be expressed with rules that end users can understand and verify. These rules are of vital importance in helping to ensure that software operates correctly. Bertrand Meyer states that the presence of these rules in a routine should be

From page 140, all that can be ascertained is:

(1) that the stimulus/response rule form of expression relates to the basic construct of event diagrams;

(2) stimulus/response rules can be expressed rigorously but can also be translated into an English statement that end users can validate; and

(3) each operation has its own context-independent precondition and postcondition which should be expressed with rules that end users can understand and verify.

Once more, there is nothing on this page of Martin that speaks to any of the features of independent claim 1. Specifically, there is no teaching or suggestion on this page of Martin regarding units of work, identifying a point in a unit of work where state integrity is to be verified, a unit of work having a plurality of participants, obtaining rules for each participant in a unit of work in response to a determination that the unit of work is to be completed, or running the rules obtained for each of the participants to verify the integrity of an application state according to the plurality of participants in response to obtaining the rules.

Page 141 is as follows:

viewed as a contract that binds the routine and its callers [1]. It is, in effect, a contract relating to software reliability.

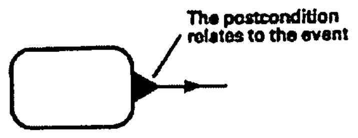
Operation precondition rules express those constraints under which an operation will perform correctly. The operation cannot go ahead unless these constraints are satisfied. Operation precondition rules have the form

```
Order a Product
    ONLY IF there is an authorized Supplier offering this Product

Promote Staff Employee to Manager
    ONLY IF Employee has a Staff position
        and Employee is not a Manager
```

In contrast, the *operation postcondition rule*, in effect, guarantees the results. It says that when the operation is executed, certain state changes will occur. Operation postcondition rules have the form

```
Order Product from supplier IS CORRECTLY COMPLETED
    ONLY IF the Product Order for supplier is created

Promote Staff Employee to Manager IS CORRECTLY COMPLETED
    ONLY IF Employee is not in a Staff position
        and Employee is a Manager
```

Events are those state changes to which a system must react. Therefore, *events* (represented by a black triangle), reflect an aspect of an operation postcondition rule. When that aspect is true, the event occurs:



CONDITIONS FOR EXECUTING AN OPERATION

Before an operation can be executed, two things must be true: its precondition, which is *independent* of the particular application, and its control condition, which is *dependent* of the application. Both of these are expressed as rules.

The user of an OO-CASE tool should be able to point at the control condition diamond or at the operation box and display the control condition rules or the operation precondition rules.

From page 141 of Martin, all that can be ascertained is:

(1) operation precondition rules express those constraints under which an operation will perform correctly and the operation cannot go ahead unless those constraints are satisfied;

(2) operation postcondition rules guarantee the results by stating that certain state changes will occur when the operation is executed;

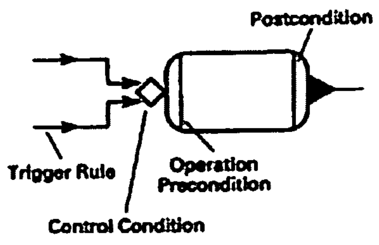
(3) events are those state changes to which a system must react and thus, events reflect an aspect of an operation postcondition rule;

(4) before an operation can be executed, its precondition, which is independent of the particular application, and its control condition, which is dependent of the application, must be true; and

(5) both the precondition and control condition are expressed as rules.

Once more, there is nothing on this page of Martin that speaks to any of the features of independent claim 1. While Martin teaches general concepts of various types of rules, there is nothing in Martin that teaches the specific features of claim 1. Specifically, there is no teaching or suggestion on this page of Martin regarding units of work, identifying a point in a unit of work where state integrity is to be verified, a unit of work having a plurality of participants, obtaining rules for each participant in a unit of work in response to a determination that the unit of work is to be completed, or running the rules obtained for each of the participants to verify the integrity of an application state according to the plurality of participants in response to obtaining the rules.

Page 142 is as follows:



The control condition relates to the trigger rules and is often designed to achieve correctness of the precondition.

These rules, displayed in windows on the event diagram, should reflect how the business people want to run the business. Business policies are explicitly stated.

EXECUTABLE DIAGRAMS

The event diagram, including statements of rules to express preconditions, triggers, control conditions, and the calculations or transformations done in an operation, is an executable diagram, that is, it can be converted to program code. The operation itself may be expressed declaratively or in a form that can drive a code generator.

> An event diagram with rules is executable.
>
> The diagrams of traditional structured analysis are not executable.

The most useful form of an OO-CASE tool is one that generates code immediately from diagrams that are executable. The code can be immediately executed to see whether it is doing what the designer intended—rather like a spreadsheet tool. When you create the columns, rows, and calculations of a spreadsheet, you can immediately run it, then quickly change it, and rerun it. You can experiment with it, adjusting its design.

An OO model is likely to be much more complex than a spreadsheet but the tool should make it equally interactive. You should be able to build it and immediately run it, add instances of class data, and observe its behavior, grow it, and change it quickly. You should be able to experiment with your design as you build it.

At first, the code will be generated interpretively and need not be machine-efficient. Later, when the design works well, it can be compiled and perhaps redesigned for maximum efficiency.

The ability to run the design immediately, experiment with it, and change it, greatly increases creativity. It expands our ability to invent interesting software.

From page 142, all that can be ascertained is:

(1) control conditions relate to trigger rules and are often designed to achieve correctness of a precondition;

(2) the event diagram is an executable diagram, i.e. it can be converted to program code;

(3) the diagrams of traditional structured analysis are not executable;

(4) OO-CASE tools may generate code immediately from diagrams that are executable; and

(5) An OO model may be more complex than a spreadsheet but should be able to be created quickly using OO-CASE tools, immediately run it, add instances of class data, observe its behavior, grow it, and change it quickly.

Yet again, there is nothing on this page of Martin that speaks to any of the features of independent claim 1. There is not even so much as the mention of a unit of work, participants of a unit of work, rules associated with a unit of work, let alone the specific features recited in claim 1, on any of pages 133-142 of Martin, despite the allegations made by the Examiner.

Page 143 of Martin reads as follows:

signed to

flect how
ly stated.

rules to
nditions,
ne in an
am code.
n drive a

tes code
nediately
er like a
ons of a
n it. You

t but the
immedi-
w it, and
ou build

nachine-
perhaps

it, and
interest-

RULES ATTACHED TO OTHER OO DIAGRAMS

Rules can be linked to other types of diagrams as well as event diagrams.

> Rules can be attached to any of the diagrams in the previous chapters.

Examples of rules associated with the diagrams of OO analysis are shown in Box 10.3.

Rules can be divided into two types—object-state rules and object-behavior rules. Object-state rules are identified in object-structure analysis. They are associated with diagrams, such as the data-structure diagram, object-relationship diagram, or composed-of diagram. Object-behavior rules are identified in object-behavior analysis. Most are associated with the event diagram; some are associated with the state-transition diagram.

Figure 10.1 shows a variety of OO diagrams with rules linked to them, as they should be in an OO-CASE tool. The rules may be expressed both in formal notation and easy-to-read English, so that either may be displayed.

Figure 10.2 shows a rule window linked to an event diagram in the IntelliCorp Object Management Workbench (OMW). A variety of such rule windows can be connected to the diagrams this tool uses. With a rule editor the analyst builds rules in English and at the same time generates executable code for the rules. The analyst can immediately run his model, animate it, and generate spreadsheets of values from it. He can thus immediately test and modify the model. The business people using the tool, possibly in facilitated workshops (discussed later), can try out different business policies and observe their effect. They can thus redesign business processes and observe the effects of the redesign.

THE PROGRAMMER AS LAWYER

The English expression of rules precise enough for code generation is formal and sometimes difficult for business people to read. The analyst should therefore translate the executable rules into easy-to-read English.

The bottom box of the Rule Editor Probe window in Fig. 10.2 contains the formal English. The box above, labeled "Meaning," contains the analyst's easy-to-read version of the same rule. The "Comment" box above this may contain comments about who wants the business policy that the rule represents, or why it is used.

Figure 10.3 shows three more rule windows that are linked to appropriate items in OO diagrams. The red item in each diagram is a rule in formal English that is created with the tool's rule-builder. As the Rule Editor assists the analyst in building the rule it simultaneously generates executable code. The analyst can compose an easy-to-read version of the rule in the box labeled "Meaning."

From page 143, all that can be ascertained is:

(1) rules can be linked to other types of diagrams as well as event diagrams;

(2) rules can be divided into object state rules and object behavior rules;

(3) object state rules are identified in object-structure analysis and are associated with diagrams, such as the data-structure diagram, object-relationship diagram, or composed-of diagram;

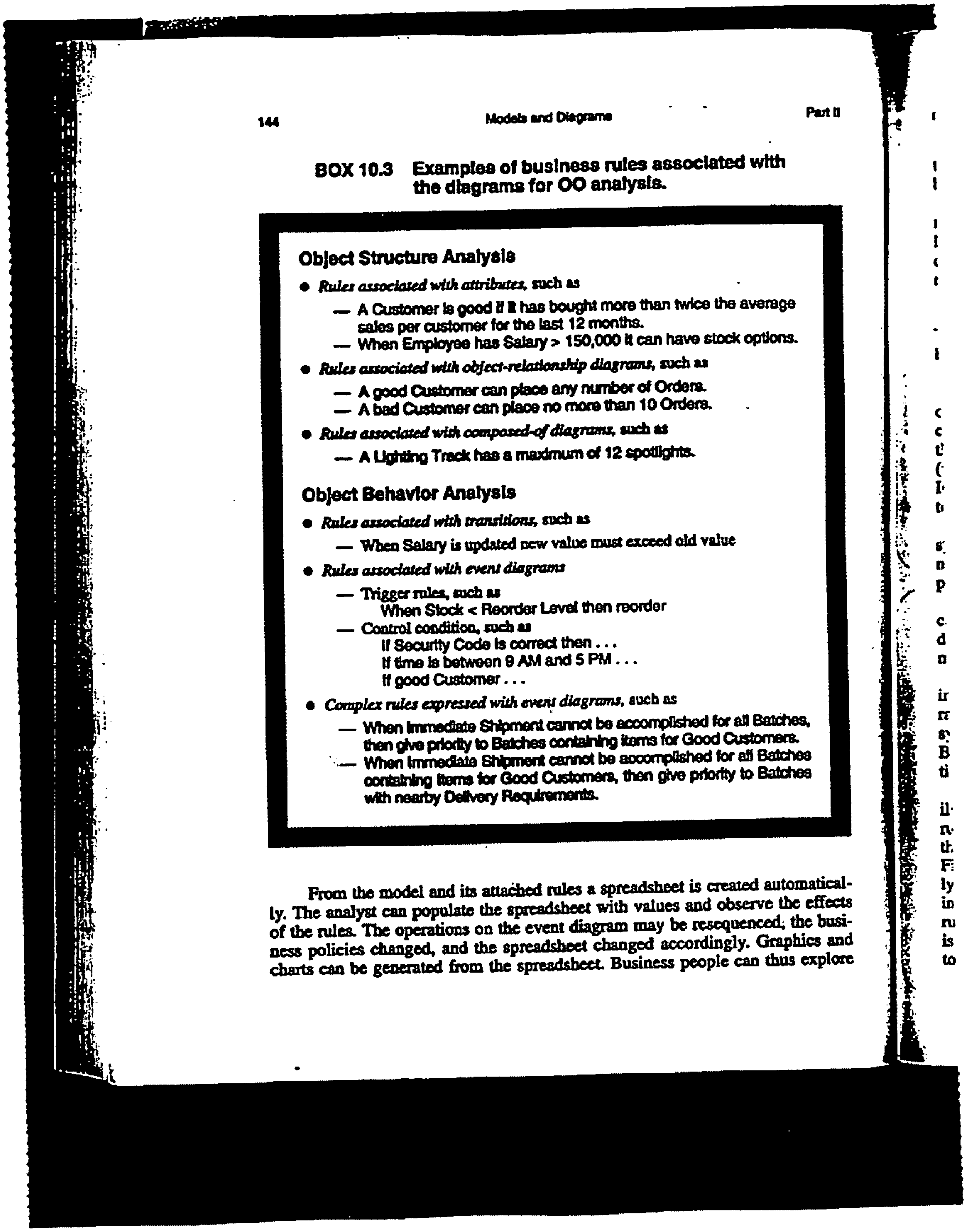
(4) object-behavior rules are identified in object-behavior analysis and are associated with event diagrams and some are associated with state-transition diagrams;

(5) rule windows may be linked to an event diagram, which display the rules associated with the boxes and symbols of the event diagram; and

(6) the analyst should translate executable rules into easy-to-read English.

Where is the teaching or suggestion of units of work, participants, rules for participants, etc.? Once more, there simply is nothing on this page of Martin that speaks to any of these features or the specific features of independent claim 1. Specifically, there is no teaching or suggestion on this page of Martin regarding units of work, identifying a point in a unit of work where state integrity is to be verified, a unit of work having a plurality of participants, obtaining rules for each participant in a unit of work in response to a determination that the unit of work is to be completed, or running the rules obtained for each of the participants to verify the integrity of an application state according to the plurality of participants in response to obtaining the rules.

Page 144 of Martin is as follows:



BOX 10.3 Examples of business rules associated with the diagrams for OO analysis.

Object Structure Analysis

- *Rules associated with attributes,* such as
  - — A Customer is good if it has bought more than twice the average sales per customer for the last 12 months.
  - — When Employee has Salary > 150,000 it can have stock options.
- *Rules associated with object-relationship diagrams,* such as
  - — A good Customer can place any number of Orders.
  - — A bad Customer can place no more than 10 Orders.
- *Rules associated with composed-of diagrams,* such as
  - — A Lighting Track has a maximum of 12 spotlights.

Object Behavior Analysis

- *Rules associated with transitions,* such as
  - — When Salary is updated new value must exceed old value
- *Rules associated with event diagrams*
  - — Trigger rules, such as
    When Stock < Reorder Level then reorder
  - — Control condition, such as
    If Security Code is correct then . . .
    If time is between 9 AM and 5 PM . . .
    If good Customer . . .
- *Complex rules expressed with event diagrams,* such as
  - — When Immediate Shipment cannot be accomplished for all Batches, then give priority to Batches containing items for Good Customers.
  - — When Immediate Shipment cannot be accomplished for all Batches containing items for Good Customers, then give priority to Batches with nearby Delivery Requirements.

From the model and its attached rules a spreadsheet is created automatically. The analyst can populate the spreadsheet with values and observe the effects of the rules. The operations on the event diagram may be resequenced, the business policies changed, and the spreadsheet changed accordingly. Graphics and charts can be generated from the spreadsheet. Business people can thus explore

From page 144 of Martin, all that can be ascertained is:

(1) examples of object structure analysis include rules associated with attributes, rules associated with object-relationship diagrams, and rules associated with composed-of diagrams;

(2) examples of object behavior analysis include rules associated with transitions, rules associated with event diagrams, and complex rules expressed with event diagrams;

(3) from a model and its attached rules, a spreadsheet is created automatically;

(4) the analyst can populate the spreadsheet and see the effects of the rules; and

(5) the operations on the event diagram may be resequenced, the business policies changed, the spreadsheet changed accordingly, graphics and charts may be generated from the spreadsheet, etc.

As with all of the previous pages, there is nothing on this page of Martin that speaks to any of the features of independent claim 1. Specifically, there is no teaching or suggestion on this page of Martin regarding units of work, identifying a point in a unit of work where state integrity is to be verified, a unit of work having a plurality of participants, obtaining rules for each participant in a unit of work in response to a determination that the unit of work is to be completed, or running the rules obtained for each of the participants to verify the integrity of an application state according to the plurality of participants in response to obtaining the rules.

The Examiner also makes reference to page 147 of Martin which illustrates the use of an integrity rule in a state transition diagram as follows:

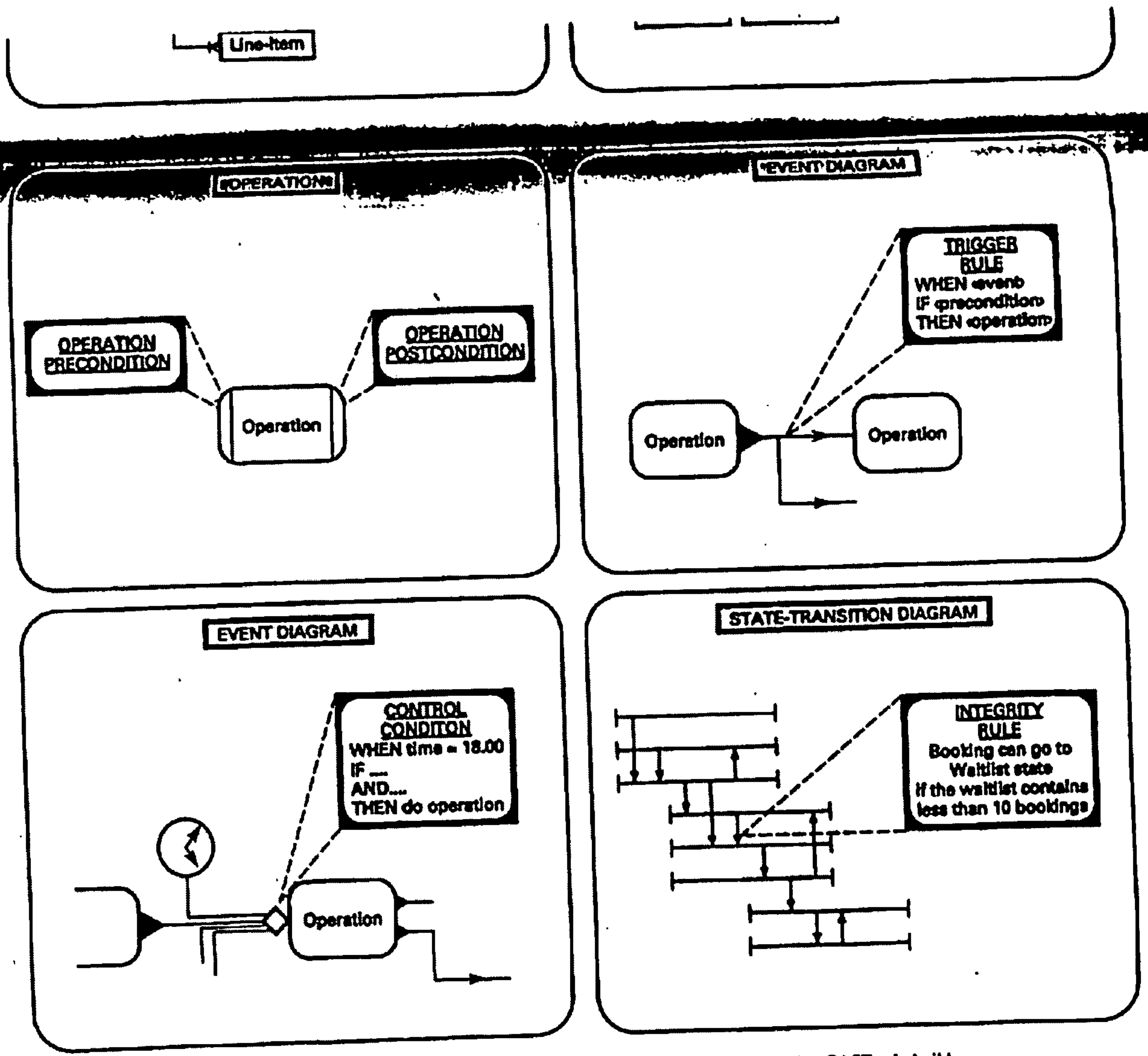


Figure 10.1 Rules linked to OO-CASE diagrams. The formal English which a CASE rule-builder creates is less easy to read than some of these illustrations. The analyst supplements the formal English with easy-to-read English to aid communication with business people.

While this page of Martin shows an integrity rule in a diagram, and diagrams are allegedly executable, there still is no teaching or suggestion in Martin regarding the specific features of claim 1. That is, where in Figure 10.1 on page 147 is it taught to identify a point in a unit of work where application state integrity is to be verified, wherein the unit of work includes a plurality of participants? There is no teaching on page 147, or any of the other pages of Martin, regarding a unit of work having a plurality of participants and identifying a point in such a unit of work where integrity is to be verified. Furthermore, where on page 147 is it taught to obtain rules for each of the plurality of participants in the unit of work in response to a determination that the unit of work is to be completed? Where on page 147 is there any mention of running rules obtained for each of the participants to verify the integrity of an application state according to the plurality of participants in response to obtaining the rules for the participants? There simply is no teaching or suggest regarding these features in Martin.

The same page-by-page analysis shown above can be performed with each of the pending claims with the result being that Martin simply does not teach or suggest the features of the claims. At most, Martin teaches integrity rules, executable diagrams, and use of integrity rules with executable diagrams. The specific features of the present invention, with regard to any of the pending claims, claim 1 merely being an example, are not taught, suggested, or even hinted at by the Martin reference. The only way that the Examiner can allegedly read the teachings of Martin on the present claims is by first generalizing the claimed invention so that many of the specific features are not addressed, and then using hindsight to assert that one of ordinary skill in the art would somehow, from the teachings of pages 133-144 and 147, come up with Appellants' exact claimed invention. Such erroneous examination of the claims stems from the Examiner's inability to take an unbiased approach to examining the claims.

VI. <u>Examiner's Response to Arguments</u>

Undoubtedly, based on the Examiner's statements in the Examiner's Answer, the Examiner would respond to the above arguments in the same manner as he did in his Examiner's Answer by saying that they are "allegations of patentability without technical merit" or that Appellant has "failed to provide technical reasoning with support from the Specification on why the claimed invention differs from the prior art of record." However, it does not take an in-depth analysis of Martin to see that, other than merely teaching integrity rules and using them with executable diagrams, there is nothing in Martin that even remotely hints at the features of the present claims. This is clear from the page-by-page analysis provided above which clearly supports Appellants arguments that have been asserted since the beginning of prosecution of the application.

Furthermore, while the Examiner may believe that such "non-technical" arguments are a "lazy" approach to overcoming the rejections (see the Examiner's Answer pages 8 and 26), Appellants have clearly shown for all features of the claims that the Martin reference does not teach any of the features of the present invention. Since Martin simply does not teach or suggest the features, and the Examiner has not established a prima facie case of obviousness by pointing to explicit teachings and suggestions where the features of the present claims are taught or suggested, Appellants need not go into the "nuts and bolts" of how the mechanisms of the Martin reference operate to explain what is apparent on the face of the reference, i.e. Martin does not teach or suggest the features of the present invention. Appellants have addressed those sections of the Martin reference that the Examiner believes teaches or suggests the features of the present invention and have shown that these sections, in actuality, do not teach or suggest anything in the present claims. That is sufficient for overcoming a rejection of obviousness.

Much of the Examiner's argument rests on the Examiner's interpretation of terms in the claim, interpretations which are devoid of any attempt to reconcile them with the present specification and are clearly obtained from a hindsight notion of attempting to fit the present claims into the mold of Martin. For example, the Examiner repeatedly states that Appellants have not clearly and concisely traversed the interpretation of "unit of work" given by the Examiner. However, Appellants have clearly stated in the Response filed August 8, 2001 that an example definition of "unit of work" obtained from the specification page 11, lines 21-23, which the Examiner conveniently disregards in his statements in the Examiner's Answer (see page 14, Examiner's Response and page 22, Examiner's Response), is "representations of pieces of business work which define each business context in which they are carried out" with an example being shown in Figure 3. While a unit of work may have rules associated with it and may even be performed using methods, they are not in themselves simply rules and methods as the Examiner alleges. This conclusion is clear from the example shown in Figure 3.

Regardless of this, the Examiner continues to interpret this term to be methods/rules.

Thus, even when the Examiner asks for clarification of a term, and Appellants give an example definition from the specification, the Examiner still refuses to change his interpretation of the term. This refusal can only be because it would require the Examiner to no longer rely on the Examiner's reference and would force the Examiner to actually find the features of the present claims, rather than some generalization of the present invention.

Similarly, the Examiner continues to interpret "participants" to be methods despite Appellants example definition from the specification page 11, lines 25-27 of participants being elements of a unit of work that are modified by the processing carried out in association with the unit of work. Methods are not modified by the processing carried out in association with a unit of work. The Examiner is not attempting to examine the claim features to determine if they read on the prior art, but rather is attempting to figure out how he can make the prior art seem like the claimed features. Even at this, the Examiner fails since the allegations made by the Examiner regarding the Martin reference simply are not supported as set forth above in the page-by-page analysis of Martin.

It is not necessary to address each of the Examiner's Responses to Appellants arguments with regard to each grouping of the claims since all of the Examiner's Response are based in the same erroneous interpretation of Martin as discussed above. Suffice it to say, the claims of each of the groupings are allowable over the alleged combination of references for the reasons set forth in Appellants' Appeal Brief.

VII. <u>Other Cited References</u>

The other cited references, i.e. Code Complete and the IBM Computer Dictionary, are superfluous and provide no additional support for the Examiner's allegations. Even when combined with Martin, they do not rise to the level of teaching the features of any one of the pending claims. Thus, it is not necessary to address these references individually.

VIII. Conclusion

In view of the above, Appellants respectfully submit that all of claims 1-29 define over the prior art of record, Martin, Code Complete, and the IBM Computer Dictionary. Appellants therefore, request that the Board of Patent Appeals and Interferences overturn the rejection of claims 1-29 under 35 U.S.C. § 103(a).

Respectfully submitted,

Stephen J. Walder, Jr.

Stephen J. Walder, Jr.
Reg. No. 41,534
Carstens, Yee & Cahoon, LLP
PO Box 802334
Dallas, TX 75380
(972) 367-2001
Attorney for Appellants